CDF/DOC/STATISTICS/PUBLIC/8128
Version 2.4
October 24, 2007

Sensitivity, Exclusion and Discovery with Small Signals, Large Backgrounds, and Large Systematic Uncertainties

Tom Junk

Fermilab

Abstract

To exclude a possible signal, or to compute the significance of one that is observed, the measured data must be compared with the predictions of a model which includes new physics, and also a model which does not, to see which of the two (if either) can be excluded, and at what confidence level. Typically, these two models are only imperfectly specified. Uncertainties in the rates and distributions of Standard Model background processes are always present, and can be larger than the predicted signals. Often an uncertain prediction can be constrained by a subsidiary measurement, for example, by counting events in a control region separate from the signal sample. In this case, the uncertainties in the modeling are at least partly, and often mostly, statistical in nature. There remain systematic uncertainties in the extrapolation (or interpolation) from a control region to the signal region, and there may remain uncertainties in the prediction of the kinematic distributions of events in the signal region. Monte Carlo (or data) statistical uncertainties in each bin of a histogram provide an additional source of uncertainty. A limit calculator is described which includes the effects of all of these sources of uncertainty, based on the CL_s procedure used at LEP. This note updates CDF 6525, and describes a program to compute limits using `ROOT`.

1 Introduction

A search for new physics at the Tevatron typically involves collecting data in one or more histograms, binned in variables designed to separate events produced by a particular signal

process from those produced by a set of background processes. Various signal hypotheses may be tested, parameterized by masses, cross sections, branching ratios, and possibly other parameters. The signals and backgrounds too may depend on a variety of parameters which are not of primary interest but which are needed for the measurement – examples include efficiencies, acceptances, integrated luminosity, and background production cross sections. The parameters which describe the signal and background processes which are not being measured or constrained by the analysis are called "nuisance parameters." Their values are needed in order to extract measurements of, or limits on, the parameters of interest, and uncertainty in their values usually results in reduced sensitivity to the parameters of interest. The systematic errors on observables are parameterized in terms of these nuisance parameters.

In the CL_s limit-setting technique used at LEP [10], the data are compared against two models at a time [2]. One is the null hypothesis, which asserts that the Standard Model describes the data, while the other is the signal+background hypothesis, which asserts that the data are modeled by SM physics plus one or more processes not included in the SM. Models of new physics are tested one at a time against the Standard Model, using the data to decide which is preferred. Many other techniques, such as Bayesian techniques and Cousins-Feldman, test a large class of possible models at once, to see which of a large number of models is preferred. The main objection to these techniques is that the limit or discovery significance of testing a particular model with the data depends on the space of models of new physics that is considered. In Bayesian techniques, the space of possible models together with their weights is encoded in the prior, while the Cousins-Feldman construction requires exploring an entire model space. For some models of new physics, the model space may not be fully explored theoretically, or may be of too high a dimension to study fully.

Model predictions always have uncertainties associated with them, and these uncertainties arise from a variety of sources. To compute discovery or exclusion signficances, the uncertainties in the model predictions must be taken into account; uncertainty in model predictions allows models to be more compatible with any observed data. Choosing which of two models is preferred by the data is made more difficult if the models have degrees of freedom which allow them to accomodate the data by adjusting their nuisance parameters appropriately. Estimations of the sensitivity of an analysis depend crucially on controlling the systematic uncertainties of the model predictions, as well as on collecting large data samples. For Higgs searches at CDF, for example, with large data samples, the systematic effects of controlling the model predictions become important compared to the statistical fluctuations, and must be handled with care.

Typical model uncertainties in histograms of predicted event counts from various processes are rate uncertainties and shape uncertainties. The parameters describing our uncertainty in rates and shapes may be constrained by statistically-limited subsidiary measurements, or they may have *a priori* constraints from theory or from long-past measurements for which the data are no longer available but a measured parameter is reported in a publication with a (possibly

asymmetric) error bar. Monte Carlo (or data) statistical uncertainties are present in each bin of a histogram used as part of a model.

A typical analysis on CDF may fit a function to a data histogram, using, for example, mass sidebands and a background fit function to predict the background underneath a signal peak. Alternatively, events in a separate histogram, collected with different selection requirements, may be used to normalize one or more background contributions in the signal region. Limits are calculated in the CL_s technique by generating pseudoexperiments, randomly varied within the statistical and systematic uncertainties, and comparing the distribution of a test statistic, such as the logarithm of the likelihood ratio, of the data against distributions obtained assuming just the null hypothesis or the hypothesis that new physics is present. It is important to treat each pseudoexperiment in the same way as the data are treated – hence sideband fits or subsidiary experiments must be also simulated and fit, and backgrounds projected into the signal region. In this manner, a large class of "systematic" uncertainties can be treated in a frequentist manner [8], although there may remain additional sources of nonstatistical systematic uncertainty.

This note is an extension of the procedure considered in CDF 6525, and a program is descibed here, which runs in `ROOT`, which considers a broader class of possible systematic uncertainties. The handling of shape uncertainties is also improved, using a histogram interpolation procedure [6] pioneered by the DELPHI experiment at CERN. I'd like to thank Tom Wright for contributions to the rate limit calculator in the software described in this note.

2 Procedure for Computing Limits

2.1 The Test Statistic

As described in [10] and CDF 6525, a choice of a test statistic which is usually optimal is the likelihood ratio

$$Q = \frac{P(\text{data}|H_1)}{P(\text{data}|H_0)} \tag{1}$$

where H_1 is a model including new physics, and H_0 is the null hypothesis, that new physics is absent. There is no requirement that H_1 predicts more data events than H_0. In fact, for some searches, such as the search for charged Higgs bosons in top decays, observing fewer events than predicted is the sign of new physics. Other new physics processes may interfere destructively with the SM background, as is the case with some Z' searches for some mass ranges. CDF 6525 uses expressions assuming that signal processes add incoherently to background processes and thus is not applicable to these searches. Both H_1 and H_0 predict data counts in each bin of the data histograms to be studied, and these predictions are usually sums from distinct

processes, added incoherently. Processes that add coherently should be treated as a single process. For example, in a Z' search, $(Z/\gamma^*/Z')$ can be considered just one process if interference is important, while fakes are another process.

At issue is the fact that $P(\text{data}|H_1)$ and $P(\text{data}|H_0)$ are sensitive to the fact that H_1 and H_0 may be poorly specified. The approach taken here is that used by analysis techniques which find the best-fit model to the data, where these probabilites are maximized over the space of of possible values of the nuisance parameters. This procedure is usually called the "profile likelihood" technique, although its interpretation and use are different here. This topic is discussed below, in Section 3.3.

The subject of finding the $P(\text{data}|H_1)$ which is maximal over variations of nuisance parameters is discussed in CDF 7904 [9]. The uncertainties which are considered in that chisquared function are rate uncertainties, histogram shape uncertainties, bin-by-bin Poisson statistical uncertainties in the model predictions, and correlations between nuisance parameters. The χ^2 calculator of CDF 7904 has been updated to compute a global χ^2 summed over several histograms which share correlated systematic uncertainties, and to allow for 2D histograms with shape errors. Furthermore, the treatment of asymmetric errors has been improved, as is described in Section 2.2.

The χ^2 function described in CDF 7904 (inspired by T. Devlin's CDF 3126) is based on the likelihood function, and in fact, the test statistic of Equation 1 is given by

$$-2\ln Q = \chi^2(\text{data}|H_1) - \chi^2(\text{data}|H_0) = \Delta\chi^2 \tag{2}$$

The minimization of χ^2 over the nuisance parameters must be done separately for the two hypotheses H_1 and H_0.

2.2 Asymmetric Errors

The χ^2 function of CDF 7904 uses systematic uncertainties on rates that are parameterized by a symmetric, multiplicative scale factor. The rate for a specific process prediction in a particular bin is sensitive to a particular nuisance parameter

$$r_{ij}^{\text{varied}} = r_{ij}^{\text{central}} \prod_k (1 + s_k f_{kj}) \tag{3}$$

where i runs over histogram bins, j indexes which model component is being considered, and k indexes the nuisance parameters s_k which are Gaussian distributed around zero with unit width (except when truncated to keep the model prediction non-negative for all components of the model). The quantity f_{kj} is the fractional uncertainty on the rate of the r_{ij} histogram due to the kth nuisance parameter. Often, f will have two values – one positive and one negative.

An interesting discussion of asymmetric uncertainties is given in Reference [11]. The approach taken here is to use Model 2 of [11], to parameterize the effect of a Gaussian-distributed nuisance parameter on a physical quantity with a quadratic function. This procedure makes the PDF's of the physical quantities continuous. One conundrum which arises in a complicated analysis is that a particular nuisance parameter (like the jet energy scale) can have a symmetric impact on, say, a background contribution, but an asymmetric impact on a signal contribution. Rather than generating pseudoexperiments with a discontinuous PDF for a nuisance parameter (like the signal efficiency), it is better to just parameterize the efficiency quadratically on a smoothly varying nuisance parameter. The variation considered here is given by

$$r_{ij}^{\text{varied}} = r_{ij}^{\text{central}} \prod_k \left(1 + s_k \left(\frac{f_{kj}^+ - f_{kj}^-}{2}\right) + s_k^2 \left(\frac{f_{kj}^+ + f_{kj}^-}{2}\right)\right) \tag{4}$$

The quantity f_{kj}^+ is the fractional change in the rate for process j when the nuisance parameter s_k is positive one unit. It may be positive or negative. The quantity f_{jk}^- is the fractional change in the rate of process j when the nuisance paramter s_k is negative one unit. For symmetric uncertainties, $f_{kj}^+ = -f_{kj}^-$.

2.3 Histogram Interpolation and Extrapolation

Because the distributions of reconstructed variables are uncertain in addition to their rates, the limit calculator must know about such uncertainties in order to provide the most honest limits. Such uncertainties, such as the jet energy scale error shifting the background template in the m_{jj} distribution can easily mask a small signal on the steeply falling edge of the background distribution.

Histograms are interpolated within their shape uncertainties on each pseudoexperiment. Histograms may be interpolated "horizontally" or "vertically". Vertical interpolation is just a linear interpolation of bin contents, with the restriction that bin contents cannot go negative (the bin contents are simply set to zero in the software described in this note). Horizontal interpolation uses `PVMORPH` [6]. In the software, the histogram interpolation style can be specified as horizontal or vertical on a channel-by-channel basis. Horizontal interpolation is more appropriate when the shape variations shift the values of the variables being histogrammed. An example is the above jet energy scale uncertainty shifting m_{jj} up and down. Vertical interpolation may be more appropriate for neural net output histograms, where horizontal interpolation may produce spurious third peaks when interpolating two histograms each with peaks at 0 and 1 and nothing in between. Two-dimensional histogram interpolation is possible both horizontally and vertically.

In general, histogram interpolation is a much more reliable procedure than histogram extrapolation, and so the `MINUIT` minimization procedure used to minimize the χ^2 test statistic

is not allowed to venture into a portion of nuisance parameter space which requires histogram extrapolation. It is therefore up to the user of the software to provide shape variations of several sigma. Providing $\pm 1\sigma$ shape variations will truncate the considered space of nuisance parameter values to $\pm 1\sigma$. Fortunately, a routine is provided which can also extrapolate histograms in the cases for which shape variations are available but only for relatively small excursions of the nuisance parameters. The routine is `csm_interpolate_histogram`, just called with a value of the control parameter outside of the range given by the two templates. Extrapolations must be checked by hand to make sure that they make sense, however. The software described below does not pay any attention to the contents of overflow and underflow bins, so the histograms must be binned adequately to cover not only all relevant entries, but also those of the varied shapes as well.

The program `mclimit_csm.C` now includes "compounded" interpolation, for both the horizontal and vertical interpolation styles. Compounded interpolation is necessary since a model varying more than one nuisance parameter at a time may be needed to fit the data properly. For example, fitting a Gaussian requires varying both the mean and width simultaneously, and the nonparametric interpolations supplied with this program allow fitting more general sets of parameters. Furthermore, if two nuisance parameters both shift a template histogram by the same amount in the same way, constructing a model in which both nuisance parameters take on the value $+1\sigma$ should involve shifting the template twice as much in that way. In this manner, compounding several nuisance parameters' shape variations amounts to an extrapolation.

With vertical interpolation, the variations from the central histogram due to interpolation in each of the nuisance parameters are added linearly, and no bin is allowed to go below zero. With horizontal interpolation, the horizontal variations in the cumulative histograms are added linearly, and the cumulative histogram is not allowed to "bend over backwards", which could happen if two shape variations sharpened up a peak. While neither one is allowed to go beyond its maximum range in the program, varying both simultaneously could result in sharpening up a peak beyond a delta function. This is protected against in the code by insisting that the cumulative histogram be monotonically increasing.

2.4 Equations of Constraint

In some problems, the parameterization given above of the rate of a particular process in terms of multiplicative factors computed from the nuisance parameters s_k, which may be shared among different histograms introducing correlations, still is insufficient to describe how a subsidiary measurement in one subset of the data affects the interpretation of another subset of the data. An example of this is the very common four-sector Missing-E_T vs. Iso. method of computing the non-W background contribution in top, electroweak, and Higgs analyses. In the spirit of including the three control sectors in the MET vs. Iso plot as subsidiary experiments and

treating them in a frequentist manner, it is necessary to express relations such as $D = A*C/B$, where A, B, C and D are event counts in the four regions from a particular process. The strategy suggested here is to define three separate, one-bin (or more, if a shape is of some use here), histograms, for the A, B and C regions, each with a separate nuisance parameter describing the rates found in the three regions. Typically, no external Gaussian constraint is applied to these parameters (the mechanism for removing the Gaussian constraints is described in Section 4). The nuisance parameters still express relative changes with respect to a standard rate, so that they may be treated in the program like all other nuisance parameters.

A function should be supplied describing how the nuisance parameter for region D depends on those describing the event counts in regions A, B and C. We construct a model which predicts $a(1 \pm f_a s_a)$ events in region A, $b(1 \pm f_b s_b)$ events in region B, $c(1 \pm f_c s_c)$ events in region C, and $d(1 \pm f_d s_d)$ events in region D, where $ac = bd$ and a, b, c and d are what's expected in the data (these rates will be needed for generating pseudoexperiments), and f_a, f_b, f_c and f_d are numbers of the order of 0.1 (for generating pseudoexperiments, they represent the amount by which the true mean value of these rates will be fluctuated). Then s_d can be computed in terms of s_a, s_b and s_c by requiring

$$\frac{a\,(1 + s_a f_a)\,c\,(1 + s_c f_c)}{b\,(1 + s_b f_b)} = d\,(1 + s_d f_d) \tag{5}$$

which can be solved to obtain

$$s_d = \frac{\frac{a(1+s_a f_a)c(1+s_c f_c)}{b(1+s_b f_b)} - 1}{f_d} \tag{6}$$

If no external Gaussian constraints are applied to the s's, the precise values of the f's do not matter in the χ^2 minimization procedure. The f's used for the pseudoexperiment generation may be chosen to be different from the f's used in the fit.

In the software described below, there is no implementation of an *a priori* Gaussian constraint in the chisquare function for a nuisance parameter which is computed explicitly as a function of other nuisance parameters. The other nuisance parameters' Gaussian constraints suffice in this case.

2.5 Unconstrained Fit parameters

The external Gaussian constraint can be removed in the software described below for a particular fit parameter by including the substring "UNCONSTRAINED" in the parameter's name. This feature is not particuarly useful when computing limits, but it is useful when using the package as a general fitting tool (simply use the `csm::chisquare()` method to run a single fit).

For example, when setting a limit on a signal, one might think of unconstraining the signal rate in the test hypothesis. What this does, however, is allow the test hypothesis to be identical with the null hypothesis (when the signal cross section is allowed to float down to zero). Hence, all test statistic outcomes ($-2lnQ$) will be negative or zero, since the test hypothesis will in this case always fit any data at least as well as the null hypothesis. It is useful to fit for unconstrained parameters and extract the fit parameters using `csm::getnparams`, `csm::getpname`, `csm::getparam`, and `csm::getcov`.

3 Confidence Level Calculation

CDF 7904 raises the issue that the number of degrees of freedom is ill-defined for computing χ^2 for a general counting experiment. The Gaussian χ^2 distribution, computed by routines such as CERNLIB's `PROB` function, do not apply in the more general case considered here, since the variations of each measaurement are Poisson and not Gaussian. To interpret the χ^2 values, pseudoexperiments must be generated and the observed χ^2 values must be compared against distributions obtained in different hypotheses.

3.1 Pseudoexperiments and p-Values

The confidence level for excluding H_1, given some experimental data and a null hypothesis, is given by

$$CL_{H_1} = P_{H_1}(Q \leq Q_{obs}), \tag{7}$$

the probability that Q is less than that obtained in the data, Q_{obs}, assuming the new-physics hypothesis H_1. This hypothesis is excluded at the 95% CL if $CL_{H_1} = 0.05$, and at more than the 95% CL if $CL_{H_1} < 0.05$. Using our $\Delta\chi^2$ test statistic, this can be written as

$$CL_{H_1} = P_{H_1}(\Delta\chi^2 \geq \Delta\chi^2_{obs}), \tag{8}$$

The confidence level for excluding the background hypothesis is another p-value, known in the LEP literature as $1 - CL_b$, and for consistency, we can call it $1 - CL_{H_0}$ It is defined as

$$1 - CL_{H_0} = P_{H_0}(Q \geq Q_{obs}). \tag{9}$$

or

$$1 - CL_{H_0} = P_{H_0}(\Delta\chi^2 \leq \Delta\chi^2_{obs}), \tag{10}$$

$1 - CL_{H_0}$ is the probability that the null hypothesis will give an outcome that looks at least as signal-like as the one observed. For discovery, $1 - CL_{H_0}$ is required to be no more than

2.87×10^{-7}, or twice that, depending on how one interprets what is meant by "five sigma," including just one side of a Gaussian tail or both. A "three sigma" excess is defined to be $1 - CL_b = 1.3 \times 10^{-3}$ or twice that.

The quantity

$$CL_s = \frac{P_{H_1}(\Delta\chi^2 \geq \Delta\chi^2_{obs})}{P_{H_0}(\Delta\chi^2 \geq \Delta\chi^2_{obs})} \tag{11}$$

is used by the LEP experiments because it is better behaved for exclusion than CL_{H_1} alone because it cannot be used to exclude a hypothesis to which there is no experimental sensitivity, while in the case of CL_{H_1}, 5% of those hypotheses for which there is no sensitivity will be excluded at the 95% CL. CL_s introduces overcoverage in doing this. There is a detail here in that CL_s is not exactly equal to CL_{H_1}/CL_{H_0} as they are defined above, although this is approximately true. The problem arises because of the use of $\leq$ and $\geq$ instead of $<$ and $>$ above. The definition of a p-value includes the probability of the observed outcome, and so $1 - CL_{H_0}$ is a p-value which can be used to test H_0. Unfortunately, the numerator and the denominator denominator of CL_s must also include the probability of the observed outcome, and so the definitions above with the appropriate inequalities are a precise statement of what needs to be done. The difference between using $\leq$ and $<$ becomes large for experiments with a small number of expected events and only one bin in the histogram. Splitting the data up into many bins with different signal-to-noise ratio expecations reduces the probability of any single outcome and also makes the analysis more optimal.

3.2 Ensembles

These probabilities, $P_{H_0}(Q \geq Q_{obs})$ and $P_{H_1}(Q \leq Q_{obs})$ need to be computed assuming a sample space from which the observed experiment is drawn, commonly called an "ensemble." To compute these probablilites, pseudoexperiments are drawn from this ensemble, and the pseudodata are analyzed in the same way as the real data, to compute $\Delta\chi^2$ for each possible outcome. If the hypotheses H_1 and H_0 were perfectly specified, this would consist of generating Poisson random numbers in each bin of each histogram according to the perfect predictions. We could then estimate just how probable each possible experimental outcome is. The problem is that systematic uncertainties in the models prevent us from interpreting a specific outcome of the data precisely. We know exactly what we observed in the data, but we do not know from what sample space it was drawn.

A Bayesian approach to this problem is to integrate the probability of each outcome over the values of all of the nuisance parameters, weighted by the prior belief functions for each nuisance parameter (typically Gaussians, or truncated Gaussians to keep predictions from being negative). This procedure is called "marginalization" and is in common use in Bayesian techniques and mixed Bayesian-Frequentist techniques such as the variant of CL_s used at LEP.

A frequentist appraoch to handling this problem is to go back to the source of uncertainty in a nuisance parameter and treat the variation in the nuisance parameter in a frequentist way [8]. It is in this manner that the results of subsidiary measurements, such as counting events in control samples and sidebands is treated statistically instead of systematically. Unfortunately, non-statistical uncertainties are nearly always present, and can be sometimes dominant. Uncertainties from model predictions which arise from comparing different Monte Carlos, or by asking theorists what the errors are on a calculation, usually have no statistical interpretation and correspond instead to relative amounts of belief. Even the procedure for applying the measured result of a subsidiary experiment to make a prediction of a needed parameter (like a background component) may be subject to non-statistical uncertainty. The goal here is to treat all the statistical uncertainties in a frequentist manner, by generating Poisson random numbers for event-counting processes in the main and subsidiary experiments, and by marginalizing over the remaining nuisance parameters.

3.3 Profiling and Marginilazation

One might ask if a profile likelihood is employed, which is commonly used to incorporate systematic uncertainties in limits, why then marginalization is necessary. The answer is that the profile likelihood is the test statistic, and its value is not used directly to compute the required p-values. Instead, an ensemble of pseudoexperiments needs to be generated to determine its distribution, since the structure of an analysis, with complicated sideband fits and subsidiary experiments and bins with very different values of the signal-to-background ratios may be present. No general assumption is made on the distribution of $\Delta\chi^2$ in either hypothesis, instead it must be computed anew for each model tested.

Marginalization is necessary in order to incorporate the effects of systematic uncertainty in the limits at all. If the search consists of counting events passing just one set of selection criteria, that is, the final histogram has just one bin, then minimizing and fitting does not re-order the outcomes. An experimental outcome with more observed events looks more signal-like than one with fewer[1], and all test-statistics are equivalent to the event count. Without marginalization of the probabilities of each outcome, systematic uncertainties would be ignored entirely in this case.

One might then ask if marginalization must be included, why the profile likelihood is necessary. The reason is to improve the sensitivity of a search. If we seek a small peak on top of a large and *a priori* uncertain background, then fitting the background shape and comparing the data to the fit is the most sensitive thing to do. To start with an *a priori* guess of the

[1]This is true for processes with signals that add incoherently to the background. The Likelihood-Ratio test statistic proposed here has the same ordering whether the new physics hypothesis has more or fewer events (or a mixture of relative signs in different channels or even bins of histograms).

background rate and shape with large uncertainties and to do no fit allows the marginalization procedure to assign very weak discrimination power to the events observed in the region of the bump. Another way of looking at this is that if the background rate is underestimated and the background uncertainty is underestimated, one obtains a high rate of false discoveries (no statistical procedure can protect against incorrectly estimated uncertainties). Similarly, if the background is overestimated and its uncertainty is underestimated, one obtains exclusions that are too powerful (almost all experiments come out with deficits of candidate events). Properly marginalized, we are uncertain whether a particular outcome represents an excess or a deficit, since we do not know what the right value of the background is. Fitting for the background, possibly using correlated fits with several histograms of differently selected data, allows us to constrain the background better than our *a priori* estimations.

Coverage is always defined relative to some ensemble, and because of the construction of the p-values CL_{H_1} and $1 - CL_{H_0}$, they cover exactly or have the usual overcoverage from Poisson discreteness.

3.4 Sensitivity

Before the experiment is conducted, the sensitivity should be estimated, to see if it is a worthwhile experiment to do and to decide what resources are to be devoted to it. Furthermore, the sensitivity should be used as a figure of merit to decide how to optimize the analysis. All activities of experimentalists should be devoted to improving the sensitivity of their analyses – from designing and upgrading the detector to choosing cuts and selecting among different advanced analysis techniques.

There is more than one figure of merit of course. One is the median expected CL_s value in an ensemble of experiments assuming H_0 – lower is better. Another is the median limit on the mass of a new particle, or the median limit on the cross section times branching ratio of a new particle, again in an ensemble of experiments assuming the null hypothesis (these last two are often equivalent). The median is used because of its invariance under transformations of variables. If we place limits on a coupling constant or a cross section (which is usually proportional to a coupling constant squared), then the median limit on one corresponds to the median limit on the other, while an average will be pulled to one side by the transformation.

One may also compute the amount of luminosity required for a desired median rate or mass limit, or the amount of luminosity required for a desired median $1 - CL_{H_0}$ discovery p-value. Optimizing an analysis for discovery can result in a different set of event selection requirments than optimizing for exclusion. Properly done, however, events should be collected in histograms with different bins, segregating regions of high signal-to-background ratios from those with low signal-to-background ratios. Optimizing for discovery usually involves finding the subsample

of data which has a very high signal-to-background ratios, even at the cost of acceptance and efficiency. Optimizing for exclusion usually involves improving signal acceptance at the cost of letting in more background. Separating events into high signal-to-background ratios and low signal-to-background ratios classes and combining the results gives optimal sensitivity for all cases.

One problem with estimating the sensitivity, particularly when the *a priori* systematic uncertainties on the signal acceptance and background rates are very large, is that some estimate of the acceptance and background rates must be supplied before the experiment is conducted, even if these will be fit to the data once data are collected. One must do the best to provide realistic estimations of these quantities for computation of sensitivity even if they later will not be needed.

3.5 The 5σ Problem

One of the issues that has plagued for a long time the computation of discovery p-values is how to quantify the significance of an outcome which is unlikely to happen in the null hypothesis – usually the question is just how unlikely in the null hypothesis the particular outcome or one that looks more like new physics could have happened. When excluding new physics models, 95% CL exclusion is usually the criterion chosen, and one only has to compute CL_s with enough precision to tell that an observed outcome is less probable than about 5% of the time assuming a signal is present. But forming discovery p-values, we must compute $1-CL_b$ values of the order of 1×10^{-7}. This computation involves generating of the order of 1×10^8 pseudoexperiments, just to be on the safe side. The program described here runs rather slowly, and computers are not yet fast enough to do the calculation comfortably many times. The calculation may be parallelized, but there is always competition for computer resources.

The traditional solution is to compute the probability in the tail of a χ^2 distribution if one knows the number of degrees of freedom, using the CERNLIB `PROB` function. One problem here is that the distribution of the $-2\ln Q$ test statistic is not a true chisquared distribution due to the Poisson nature of the data. If the s/b of all bins of the analysis is $\ll 1$, one can approximate the distributions as Gaussian and do away with needing to run psuedoexpeirments (although enough should be run to verify the shape of the core of the distribution).

A trick used at LEP is that the likelihood ratio Q is in fact the ratio of the PDF's evaluated at that particular Q. The inclusion of systematic uncertainties makes the argument less clear, and in this instance, where Q is a ratio of likelihoods maximized over nuisance parameters, and the values of the nuisance parameters which maximize the likelihoods in the $H1$ and $H0$ cases may be different.

No practical solution is proposed yet here. One may arbitrarily parameterize the PDF

of $-2 \ln Q$ for the two hypotheses and integrate the tails of the parameterization, although it is best to check it at least once with a very high-statistics calculation. Sometimes one must compute many such small p-values, particularly when estimating the sensitivity of a measurement. Typically one wants to know in what fraction of signal experiments one gets a result of a chosen significance, as a function of various physics parameters (such as the mass of a new particle), the integrated luminosity, acceptance and background rates, and systematic uncertainty parameterizations.

An issue comes up when the search analysis has a mixture of one or more channels or bins with a low expected s/b and a large expected event rate, which are combined with one or more bins with low backgrounds and higher s/b. The PDF of the test statistic then is a convolution of a Gaussian chisquare distribution with a discrete Poisson distribution. The inclusion of the fit to maximize the likelihood for each hypothesis further distorts the picture. It is hard to make a prediction of the form of the PDF of the test statistic without doing pseudoexperiments.

4 Available Software

This section describes a limit calculator based on the $\Delta\chi^2$ test statistic described above, and the program runs as a compiled script in `ROOT`. The inputs are `TH1` histograms and histograms of classes which inherit from `TH1`, such as 2D histograms. It is built on the χ^2 calculator described in CDF 7904, which has been extended to compute a joint chisquare over many histograms of data, so that several searches for new physics may be combined together. The code is available at `http://www.hep.uiuc.edu/home/trj/cdfstats/mclimit_csm1/index.html`, along with some examples of how to use it.

One uses this package by creating a member of the class `mclimit_csm` and creating instances of the class `csm_model` which describe the signal and null hypotheses. Instances of class `csm_model` have inside of them template histograms which are fit to the data, as well as descriptions of all of the systematic uncertainties and their correlations. Separate instances of `csm_model` are used to generate pseudoexperiments and to fit them, so that the user may study the effects of estimated central values used in the fits for backgrounds, for example, which are systematically different from the ones used to generate pseudoexperiments. No statistical technique can protect against incorrectly estimated backgrounds with underestimated systematic uncertainties, but at least the tools to minimize the impact and to study the residual effects are provided here.

A channel corresponds to one data histogram, and typically corresponds to a single analysis team's result. An example is to use the dijet mass distribution in a Higgs boson search, or a neural-net output distribution. Subsidiary experiments should be included as separate channels, although mass sidebands that are included in the same histogram and fit together are included in the same channel as the signal. A channel model, of class `csm_channel_model`, consists of

template histograms for the components to sum up to predict the event counts in each bin, as well as uncertainties in rates and shapes. A `csm_model` is a collection of `csm_channel_model`'s, along with optional constraint relationships between nuisance parameters. Most correlations in systematic uncertainties are handled simply by using the same nuisance parameter name to refer to effects on two different distributions. For example, if the jet energy scale affects the signal acceptance and the background rate and their shapes, then the nuisance parameter named after the jet energy scale should be re-used to parameterize all of those uncertainties, so that they move together in the pseudoexperiments and in the fits. Nuisance parameters with the same name are taken to be 100% correlated and nuisance parameters with different names are taken to be 0% correlated (unless an equation of constraint is supplied). Arbitrarily-correlated systematic errors can always be decomposed into 100% and 0% correlated pieces.

The data histograms are supplied, identified by their channel names. The members of class `csm_model` refer all template histograms to the channel to which they correspond.

A new feature added is to call the `genlimit` program from Joel Heinrich [12]. It is natural to have set this up within this program, since `genlimit` needs a "Bayesian ensemble" in order to compute its limits, and this ensemble is precisely the pseudoexperiment ensemble produced here used to compute p-values. The Bayesian limit routines are often quicker than the rate limit calculators which depend on CL_s. There are two choices of prior for the `genlimit` program, and the routines here set it to `corr` instead of `flat`. This can be changed in the source code if `flat` is desired.

The Bayesian ensemble is generated using the test-hypothesis pseudoexperiment model, since it is the one which is expected to list all the signals and backgrounds with all of their correlated errors. The test-hypothesis model, used to fit to the pseudodata, may in many situations, be a stripped-down model with fewer nuisance parameters to fit than are varied. The expected limits are computed by generating null-hypothesis pseudoexperiments (using the null-hypothesis pseudoexperiment model) and then interpreting them just like the data.

4.1 Member Methods of `csm_model`

```
----------------------------------------------------------------------

void csm_model::add_template(TH1 *template_hist,
                             Double_t sf,
                             Int_t nnp,
                             char* npname[],
                             Double_t *nps_low,
```

```
                                Double_t *nps_high,
                                TH1 *lowshape[],
                                Double_t *lowsigma,
                                TH1 *highshape[],
                                Double_t *highsigma,
                                Int_t pflag,
                                Int_t signalflag,
                                char *channame)
```

Adds a component of the total prediction to be compared with the data, and parameterizations of the errors on this model component. Typically each component corresponds to a separate physics process, such as ‘‘background from diboson events’’ which is to be added to the predictions of other processes to get the total prediction. All of the template histograms, shape histograms, and errors are copied into dynamically allocated storage within the csm_model class, and thus the originals do not need to persist after the add_template method is called.

template_hist	may be a Poisson or non-Poisson histogram. If this histogram comes from a Poisson subsidiary process (like MC or a subsidiary measurement), be sure that its normalization corresponds to the entries made in it. (That is, let sf do all of the scaling).
sf	scale factor to multiply template by to compare w/ data (e.g., (data_lum/MC_lum) for a MC Poisson histogram
nnp	number of nuisance parameters -- each is constrained to zero by a Gaussian of unit width. Each nuisance parameter corresponds to one entry in the nps_low, nps_high, lowshape, highshape, lowsigma, highsigma arrays below.
npname	nuisance parameter names. Correlations between systematic errors across templates are handled by labeling the separate nuisance nuisance parameters by name. If the name contains the substring "UNCONSTRAINED", then when used as a template in nullhyp and testhyp, the parameter is fit without the usual Gaussian constraint.

nps_low
nps_high — These are the f's, fractional uncertainties on the normalization (sf) due to each nuisance parameter. Fractional uncertainties may be asymmetric -- when a nuisance parameter is negative, it may have a different effect on sf than when it is positive. Typically nps_low and nps_high will have opposite signs, as opposite variations of a nuisance parameter will have opposite effects on sf -- these signs need to be input as opposite in this case. But sometimes you get the same sign of variation, in which case nps_low and nps_high may have the same sign. The relative sign is important across templates too. If one template's normalization goes up while another goes down when a nuisance parameter is fluctuated (anticorrelation), this is reflected in the relative signs of these f's.

lowshape — Histogram corresponding to a variation of a nuisance parameter in the negative direction. Used to parameterize shape uncertainty. The normalization of this histogram is important, and the same value of sf (see above) is used. A rate error witht the same name as the shape error is considered fully correlated.
Set this pointer to zero if you do not have a shape variation for this template for this nuisance parameter. If the template histogram is Poisson, then lowshape and highshape should have the same number of entries as the template histogram in order to make the interpolated histogram follow Poisson statistics too.

lowsigma — How many sigma of variation the lowshape corresponds to. (example: you may make a histogram of a variable that corresponds to changing the jet energy scale by two sigma. set this number to 2. The sign of this 2 doesn't matter).
Note: histogram extrapolation is not allowed -- the nuisance parameter this shape uncertainty corresponds to is constrained to lie between -|lowsigma| and +|highsigma|.

highshape — Same as lowshape, but for positive variations of the corresponding nuisance parameter. Set it to zero if you

```
                don't have this uncertainty evaluated.

highsigma       See the description of lowsigma.

pflag           Set to 1 if the template histogram is Poisson distributed
                and set to 0 otherwise.

signalflag      Set to 1 if this template histogram is to be scaled when
                computing rate limits and zero if not.  Typically identifies
                signal histograms from background histograms when appropriate.

channame        Name of the channel to which this template corresponds.

-------------------------------------------------------------------------------

void csm_model::add_chanmodel(csm_channel_model *chanmodel,char *channame)

  You can use the class csm_chanmodel to build a model for each channel
  separately, and add them all in to the multichannel model using this if you
  like.

-------------------------------------------------------------------------------

void csm_model::set_interpolation_style(char *channame, INTERPSTYLE )
    Sets the interpolation style for the channel named channame.
    Either CSM_INTERP_HORIZONTAL or CSM_INTERP_VERTICAL
    horizontal (csm_pvmorph) is the default if this is not called.

-------------------------------------------------------------------------------

void csm_model::add_npcons(Int_t nparinput, char **inputparnames,
                           char *outputparname,
                           Double_t (*f)(Double_t*))

  For nuisance parameters which are functions of other nuisance parameters.
  Call this once per constraint function (you may supply more than one
  constraint function, but circular references are not allowed).

nparinput       Number of nuisance parameters the one to be
                computed depends on.
```

```
inputparnames   The names of the input nuisance parameters as an
                array of pointers to strings

outputparname   The name of the constrained nuisance parameter

f               A pointer to a function which takes as its argument an array
                of nuisance parameter values which correspond in number,
                order, and name to inputparnames, and computes the resulting
                nuisance parameter named by outputparname.  When generating
                pseudoexperiments and when fitting, f is called on each
                iteration to compute outputparname as a function of the
                other nuisance parameters.
```

```
void csm_model::plotwithdata(char *channame, TH1 *datahist)

Makes a stacked histogram of the model predictions compared against the
supplied data histogram.
```

```
void csm_model::print()

Prints some debugging information to stdout.
```

```
csm_model* csm_model::Clone()

Makes an exact copy of the model.  Useful if you're using the background model
for some search as a subset of the model for signal+background.
```

```
csm_model* csm_model::add(csm_model &modeltoadd)

Includes all the channels, template histograms, and constriants of
modeltoadd to the model of which this method is a member and returns
```

```
a new model which is the union of all of these things.  Leaves the two
original models unchanged.

-------------------------------------------------------------------------

csm_model* csm_model::scale(Double_t scalefactor)

Scales all of the template histograms (really just their {\tt sf}'s)
and returns a pointer to the new scaled model.  Leaves the orignal
model alone.

-------------------------------------------------------------------------

csm_model* csm_model::scalesignal(Double_t scalefactor)

Scales only those template histograms which are identified as
``signal'' histograms and returns a pointer to the new
scaled model.  Leaves the orignal model alone.  This is ideal for
computing cross-section times branching ratio limits for new signals
which add incoherently to backgrounds (used in the
mclimit_csm::s95 routines below), but may be inappropriate when
the presence of a signal reduces the expected event count.

-------------------------------------------------------------------------

csm_model* csm_model::scale_err(Double_t scalefactor)

Works like csm_model::scale, but reduces all systematic error
components proportional to 1/sqrt(scalefactor), to allow projections
of how much luminosity is needed for a particular level of sensitivity,
assuming systematic uncertainties scale inversely with collected data.
```

4.2 Member Methods of mclimit_csm

```
-------------------------------------------------------------------------
Specifying Inputs to the limit calculator:
```

```
   void set_null_hypothesis(csm_model *null_hypothesis)
   void set_test_hypothesis(csm_model *test_hypothesis)
   void set_null_hypothesis_pe(csm_model *null_hypothesis_pe)
   void set_test_hypothesis_pe(csm_model *test_hypothesis_pe)

The null hypothesis model is fit to the data and pseudodata using the
chisquared minimization fit, as is the test hypothesis model, which is
used to form the delta-chisquared test statistic.  Separate models are
used to generate pseudoexperiments.  These may be the same as those used
in the fits, but may also be different in order to study the effects
of biases.

   void set_datahist(TH1 *datahist, char *channame)   The histograms
                             for the observed events in the experiment.

   void set_npe(Int_t npe)    Sets the number of pseudoexperiments to do.
                              The default is set in the constructor to 10000

   Int_t get_npe()            returns the value set in set_npe

------------------------------------------------------------------------
Controlling MINUIT
Each pseudoexperiment runs MINUIT twice, each using a csm instance
These parameters are passed into csm (see csm's versions of these) on
each pseudoexperiment.

void mclimit_csm::setminuitmaxcalls(Int_t maxcalls)
Int_t mclimit_csm::getminuitmaxcalls()

  The maximum number of calls to the minimization function allowed to MINUIT
  Default: 500 if you don't call the setminuitmaxcalls routine.

void mclimit_csm::setminuitstepsize(Double_t stepsize)
Double_t mclimit_csm::getminuitstepsize()

  The starting stepsize put into the MNPARM call, in units of sigma.  The default
  is 0.1 if you don't call setminuitstepsize().

void mclimit_csm::setprintflag(bool pf)
bool mclimit_csm::getprintflag()
```

```
  Turns MINUIT's printing on and off.  Default is off (false)

void mclimit_csm::setminosflag(bool mf)
bool mclimit_csm::getminosflag()

  Turns on/off MINOS.  Defaults is off (false).

------------------------------------------------------------------------
Running pseudoexperiments for a single model comparison:

   void run_pseudoexperiments()    Runs pseudoexperiments for both hypotheses
    You need to call this before accessing cls(), clb(), the test statistic
    distributions, or the discovery potential accessors.  An error message
    will be printed out if run_pseudoexperiments() has not been called since
    the last modification to the input models and before the invocation of a
    method which needs it.

  void mclimit_csm::setpxprintflag(bool pf)
  bool mclimit_csm::getpxprintflag()

  Sometimes it is convenient to print out values of the test statistic on each
  pseudoexperiment for later plotting, possibly with another program.
  These methods also switch on printing of psedoexperiment results with the
  Bayesian calculators.  The default is not to print out the values.
  Format of each line:
    nullhyp_px   testhyp_px   endl
  This output can get cluttered up if you also have MINUIT printing turned on.
------------------------------------------------------------------------
Accessing CLs and associated quantities

   Double_t cls()     Returns the value of CLs defined in the text
   Double_t clsb()    Returns CL(H1), the numerator of CLs
   Double_t clb()     Returns P(Delta chisquared >= Delta chisquared(obs)|H0),
                      the denominator of CLs.
   Double_t omclb()   "1-clb" computed as a p-value, including the probability
                      of the exact outcome observed in the data.

   Double_t ts()      The delta chisquared test statistic (=-2lnQ) computed
                      for the data histogram
```

```
------------------------------------------------------------------------
Distributions of test statistic in null hypothesis (H0) pseudoexperiments.
Useful for drawing bands on plots of expected test statistics vs.
a parameter like new particle mass or cross-section times branching ratio.

   Double_t tsbm2()  2 sigma low edge
   Double_t tsbm1()  1 sigma low edge
   Double_t tsbmed() median test statistic in null hyp pseudoexperiments
   Double_t tsbp1()  1 sigma upper edge
   Double_t tsbp2()  2 sigma upper edge

------------------------------------------------------------------------
Distributions of test statistic in null hypothesis (H1) pseudoexperiments.
Useful for drawing bands on plots of expected test statistics vs.
a parameter like new particle mass or cross-section times branching ratio.

   Double_t tssm2()  2 sigma low edge
   Double_t tssm1()  1 sigma low edge
   Double_t tssmed()  median test statistic in null hyp pseudoexperiments
   Double_t tssp1()  1 sigma upper edge
   Double_t tssp2()  2 sigma upper edge

------------------------------------------------------------------------
Distributions of expected CLs values

   Double_t clsexpbm2()  Expected cls in null hyp -- 2 sigma low edge
   Double_t clsexpbm1()  Expected cls in null hyp -- 2 sigma low edge
   Double_t clsexpbmed() Expected cls in null hyp -- median
   Double_t clsexpbp1()  Expected cls in null hyp -- 1 sigma upper edge
   Double_t clsexpbp2()  Expected cls in null hyp -- 2 sigma upper edge

   Double_t clsexpsm2()  Expected cls in test hyp -- 2 sigma low edge
   Double_t clsexpsm1()  Expected cls in test hyp -- 2 sigma low edge
   Double_t clsexpsmed() Expected cls in test hyp -- median
   Double_t clsexpsp1()  Expected cls in test hyp -- 1 sigma upper edge
   Double_t clsexpsp2()  Expected cls in test hyp -- 2 sigma upper edge

   These accessors below use the CLs definition of CLb which includes the
   probability of observing exactly the data outcome
```

```
   (subtracting it from 1 makes 1-CLb computed with these routines omit the
   probability of observing exactly the data outcome)
   Double_t clbexpsm2()    Expected clb in test hyp -- 2 sigma low edge
   Double_t clbexpsm1()    Expected clb in test hyp -- 2 sigma low edge
   Double_t clbexpsmed()   Expected clb in test hyp -- median
   Double_t clbexpsp1()    Expected clb in test hyp -- 1 sigma upper edge
   Double_t clbexpsp2()    Expected clb in test hyp -- 2 sigma upper edge

   These accessors below use the p-value definition of 1-CLb
   which includes the probability of observing exactly the data outcome
   Double_t omclbexpsm2()  Expected clb in test hyp -- 2 sigma low edge
   Double_t omclbexpsm1()  Expected clb in test hyp -- 2 sigma low edge
   Double_t omclbexpsmed() Expected clb in test hyp -- median
   Double_t omclbexpsp1()  Expected clb in test hyp -- 1 sigma upper edge
   Double_t omclbexpsp2()  Expected clb in test hyp -- 2 sigma upper edge

   These accessors below use the CLs definition of CLb which includes the
   probability of observing exactly the data outcome
   (subtracting it from 1 makes 1-CLb computed with these routines omit the
   probability of observing exactly the data outcome)
   Double_t clbexpbm2()    Expected clb in null hyp -- 2 sigma low edge
   Double_t clbexpbm1()    Expected clb in null hyp -- 2 sigma low edge
   Double_t clbexpbmed()   Expected clb in null hyp -- median
   Double_t clbexpbp1()    Expected clb in null hyp -- 1 sigma upper edge
   Double_t clbexpbp2()    Expected clb in null hyp -- 2 sigma upper edge

   These accessors below use the p-value definition of 1-CLb which includes the
   probability of observing exactly the data outcome
   Double_t omclbexpbm2()  Expected 1-clb in null hyp -- 2 sigma low edge
   Double_t omclbexpbm1()  Expected 1-clb in null hyp -- 2 sigma low edge
   Double_t omclbexpbmed() Expected 1-clb in null hyp -- median
   Double_t omclbexpbp1()  Expected 1-clb in null hyp -- 1 sigma upper edge
   Double_t omclbexpbp2()  Expected 1-clb in null hyp -- 2 sigma upper edge

------------------------------------------------------------------------
Probabilities of having excesses at various levels of significance

   Double_t p2sigmat()     Probability of a 2-sigma evidence
                           assuming test hyp. is true
   Double_t p3sigmat()     Probability of a 3-sigma evidence
```

```
                          assuming test hyp. is true
   Double_t p5sigmat()    Probability of a 5-sigma discovery
                          assuming test hyp. is true
   Double_t p2sigman()    Probability of a 2-sigma
                          evidence assuming null hyp. is true
   Double_t p3sigman()    Probability of a 3-sigma
                          evidence assuming null hyp. is true
   Double_t p5sigman()    Probability of a 5-sigma
                          discovery assuming null hyp. is true
```

Rate limit calculators -- These run CLs calculations many times inside of them in order to find the amount of signal which is just barely excluded at 95%. These work only for signals which add incoherently to the backgrounds. If new physics processes reduce the predicted event yields, the user has to do the hunting around himself for a model which is just barely excluded. Tom Wright supplied valuable code for optimizing these rate limit calculators, making them faster and more accurate.

```
   Double_t s95()     Scale factor on signal which is excluded
                      at exactly 95% CL.  Note -- this and other
                      s95 variants rely on the identification of
                      part of the test hypothesis to be scalable.
                      See the argument ‘‘signalflag’’ in
                      csm_model::add_template
   Double_t s95m2()   variation around the median expected s95
                      in the null hypothesis, -2 sigma
   Double_t s95m1()   variation around the median expected s95
                      in the null hypothesis, -1 sigma
   Double_t s95med()  median expected s95 in the background hypothesis
   Double_t s95p1()   variation around the median expected s95
                      in the null hypothesis, +1 sigma
   Double_t s95p2()   variation around the median expected s95
                      in the null hypothesis, +2 sigma

   Double_t lumi95()  Calculates the lumi needed for a median experiment
                      to exclude at 95% CL.
                      What's returned is a multiplicative factor on
                      whatever luminosity was used
                      to construct the test and null hypotheses.
```

```
   Double_t lumi3s()  calculates the lumi needed for a median experiment
                      to exclude the null hypothesis at 3 sigma assumming the
                      test hypothesis
   Double_t lumi5s()  calculates the lumi needed for a median
                      experiment to exclude the null hypothesis at 5 sigma
      (the criterion for discovery)
      assuming the test hypothesis.

------------------------------------------------------------------------
Bayesian Rate Limit Calculators, using Joel Heinrich's
genlimit program (CDF 7587)

void bayes_heinrich(Double_t beta, Double_t* sflimit, Double_t* unc)

   Bayesian limit calculator.  First arg:  credibility level:
                                  e.g., 0.95.
                               Second arg, scale factor on signal to produce the limit.
                               Third arg, uncertainty on the limit scale factor.
   As with the s95 routines above, this assumes that the signal adds incoherently to the
   background.  Requires set_test_hypothesis_pe to be called first in order to make the
   "Prior ensemble".  The size of the prior ensemble is set with set_npe()
   Also you must call the relevant set_datahist methods too.  Two prior choices
   are available in CDF 7587:  ``flat'' and ``corr''.  The default is ``corr''

void bayes_heinrich_withexpect(Double_t beta,
                               Double_t* sflimit, Double_t* unc,
               Int_t npx,
                               Double_t* sm2, Double_t* sm1,
                               Double_t* smed, Double_t* sp1,
                               Double_t* sp2)

   Call the genlimit Bayesian limit calculator, but repeat the calculation for
   npx pseudoexpeirments drawn from the null hypothesis.  The test hypothesis
   used is defined with set_test_hypothesis_pe, and the null hypothesis used
   is defined with set_null_hypothesis_pe.  Be sure to call those before using
   bayes_heinrich_withexpect.
   The routine computes the observed and expected limits.
   Arguments:  1:  beta (credibility level, e.g. 0.95)
   Argument 2:     observed limit
   Agrument 3:     error on observed limit
```

```
   Argument 4:  npx pseudoexperiments to run to compute expected limits
   Arguments 5-9: Expected limits: -2 sigma, -1 sigma, median, +1 sigma, +2 sigma


   void bayes_heinrich_coverage_check(Double_t beta,
                                      Double_t sflimit,
                                      Double_t* unc,
      Int_t npx,
                                      Double_t* falsex)

   Check the coverage -- the false exclusion rate.  Generate test_hypothesis_pe
      pseudoexperiments with sflimit times the signal rate, and see what fraction of them
      get excluded at the beta credibility level -- this fraction is falsex.  Should be
      less than or equal to 1-beta.

-------------------------------------------------------------------------
Looking at the PDF's of -2lnQ

void tshists(TH1*,TH1*)   Fills histograms with test statisic values
                          in the pseudoexperiments (be sure to call run_pseudoexperiments()
                          to fill in these histograms.
                          (you define the binning).  First histo:
                          test hypothesis, second histo:
                          null hypothesis

Looking at the distribution of s/b:

void plotlnsb(TH1 *mcb_hist, TH1 *mcs_hist, TH1 *data_hist)
     make a summed plot of ln(1+s/b) given the
     input histogram pointers.  They are filled in with summed MC
     and data (they are reset first).

Other diagnostic tools (public elements of mclimit_csm)

Some extra things that bayes_heinrich and bayes_heinrich_withexpect will compute,
if requested.  You can get a plot of the posterior PDF by specifying the range
over which it is to be evaluated and the point sample density  -- These are initialized
to zero by the constructor.  Just specify the beginning and the end of the interval and
the step size, and the bayes_posterior vector will be filled in when bayes_heinrich and
bayes_heinrich_withexpect are called.  bayes_interval_end > bayes_interval_begin and
```

```
bayes_inteval_step > 0 for bayes_posterior to be filled in.

   Double_t bayes_interval_begin
   Double_t bayes_interval_end
   Double_t bayes_interval_step

 The following two vectors are filled if the three paramters above are specified
 and bayes_heinrich or bayes_heinrich_withexpect is called.  Plot the
 contents of the first vector versus the values of the second vector.

   vector<Double_t> bayes_posterior
   vector<Double_t> bayes_posterior_points

 With bayes_heinrich_withexpect, you also can get a histogram of expected limits on the
 background-only pseudoexperiments.  Just specify the histogram to receive the entries
 (it is reset on entry into bayes_heinrich_withexpect and filled in) -- the pointer is
 initially null, and no filling occurs unless this pointer is set.

   TH1F *bayes_pseudoexperiment_limits
```

4.3 Chisquared Calculator Routines

```
void csm::set_htofit(TH1 *h, char *channame)

  This method identifies a histogram of Poisson-distributed
  data.  The histogram is cloned when the method is called,
  and the clone is deleted when the csm destructor is called
  or when set_htofit is called again.
  The overflow and underflow bins are ignored in the chisquare
  calculation.  Define a data histogram for each channel.

void csm::set_modeltofit(csm_model *model)

  This method tells the chisquared minimizer which model to fit to the
  data.  Use the add_template routines to populate a model with
  template histograms of predictions.

void csm::setminuitmaxcalls(Int_t maxcalls)
```

```
Int_t csm::getminuitmaxcalls()

  The maximum number of calls to the minimization function allowed to MINUIT
  Default: 500

void csm::setminuitstepsize(Double_t stepsize)
Double_t csm::getminuitstepsize()

  The starting stepsize put into the MNPARM call, in units of sigma.  The default
  is 0.1 if you don't call setminuitstepsize().

void csm::setprintflag(bool pf)
bool csm::getprintflag()

  Turns MINUIT's printing on and off.  The default is off (false).

void csm::setminosflag(bool mf)
bool csm::getminosflag()

  Whether or not MINOS is called after MINIMIZE in MINUIT.
  You get the most information out of this when the print flag is turned on.
  The default is not to call MINOS (set to false).

Double_t csm::chisquared()

  The chisquared calculation described here.   Runs MINUIT, and can be used
  as a general fitting algorithm.

Int_t csm::ndof()

  Very naive calculation of the number of degrees of freedom

csm_model* csm::getbestmodel()

  Returns a pointer to a model with varied nuisance parameters which
  is the best comparison with the data (minimizes chisquared).
  Be sure to call the chisquared method to do the minimization
  calculation before accessing the best model.  This method returns
  a pointer to a piece of static memory so don't delete it.  This method
  is best used with the plotwithdata method of csm_model to make a stacked
```

histogram of the best fit model with the data for diagnostic purposes.

Calls to add_template may refer to nuisance parameters multiple times
and so an internal list is made in csm of all the parameters,
identified by name. These methods access that list.

Int_t csm::getnparams()

Returns the number of independent nuisance parameters given in the
add_template calls.

Double_t csm::getparam(Int_t iparam)

Access to value of a nuisance parameters after the minimization. Be sure
to call the chisquare method before calling this method.

Double_t csm::getperror(Int_t iparam)

MINUIT's uncertainty. Don't believe it. Because of discrete behavior of
interpolated Poisson histograms, a nuisance parameter can vary by a tiny
amount and an event can flip from one bin in a model histogram to another,
changing the chisquared by a discrete amount. The chisquared function
therefore has discontinuities in it and MINUIT may get a strange derivative
if it chooses too small a finite difference.

Double_t csm::getcov(Int_t iparam, Int_t jparam)

MINUIT's covariance matrix. See the documentation for MNEMAT --
the covariance matrix should be symmetric. 0<iparam,jparam<nparams

char* csm::getpname(Int_t iparam)

The corresponding parameter's name.

csm::~csm()

Be sure to delete your instance of csm before setting up another chisquared
calculation. There is not facility to edit an existing list of template
histograms -- in order to change a chisquared calculation, the csm instance
should be deleted and the setup repeated. One exception to this -- you can

```
  call set_htofit again to find the chisquared of a new data histogram without
  rebuilding the list of model templates.  This is designed for convenience
  when running pseudoexperiments.
```

4.4 Histogram Interpolation and Extrapolation

```
void csm_interpolate_histogram(TH1 *h1, Double_t x1,
                               TH1 *h2, Double_t x2,
                               TH1 *h3, Double_t x3,
                               INTERPSTYLE istye)

Interpolates/extrapolates 1D and 2D histograms.  h1 and h2 are the
input histograms to interpolate, and x1 and x2 are parameters corresponding
to h1 and h2.  x3 is the input parameter corresponding to the desired
interpolated (extrapolated) histogram h3 (output).  The chisquared minimizer
is designed not to extrapolate a histogram in its minimization procedure --
the nuisance paramters are bounded so that x3 stays between x1 and x2.
Users of these routines are encouraged to supply several-sigma-varied
histograms in the model templates and if these are not available from
fluctuated Monte Carlos, then this routine can be used by hand to extrapolate
the histograms.  A warning message is printed in this case to the effect
that histogram extrapolation is attempted, and the resulting histogram
should be inspected and validated.  Overflow and underflow
bins are not included in the interpolation, and are, in general ignored
by this suite of routines.

If istyle = CSM_INTERP_HORIZONTAL, then template morphing is used.  If
   istyle = CSM_INTERP_VERTICAL, then the contents are linearly interpolated
                bin-by-bin.  The vertical option is best for histograms in which
                neighboring bins do not have much to do with each other, and
                also for histograms of neural net outputs.

void csm_interpolate_histogram2(TH1* central, Double_t paramcentral,
                                TH1* varied, Double_t paramvaried,
                                TH1* startshape,
                                TH1* outshape,
                                Double_t param,
```

```
                                INTERPSTYLE istyle)

Version to be used with cascading shape errors -- needs a central shape, a varied shape,
and a shape to apply the variations to (which may not be either of the above, but the result
of a different shape variation) startshape.  The output is outshape.
Histograms may be 1D or 2D.
```

References

[1] S. Baker and R. D. Cousins, 'Clarification of the Use of Chi-Square and Likelihood Functions in Fits to Histograms' Nucl. Instrum. Meth. **A221** 437 (1984).

[2] L. Lyons, 'Selecting Between Two Hypotheses', OUNP-99-12 (1999).

[3] L. Demortier and L. Lyons, CDF Note 5776 "Everything you always wanted to know about pulls", (2002).

[4] T. Devlin, CDF note 3126, 'Correlations from Systematic Corrections to Poisson-Distributed data in Log-Likelihood Functions' (1999).

[5] R. Barlow and C. Beeston, 'Fitting Using Finite Monte Carlo Samples', Comput. Phys. Commun. **77** 219-228 (1993).
See also the `HBOOK` Reference Manual, CERN Long Writeup Y250 (1995).

[6] A. L. Read, 'Linear interpolation of histograms', Nucl. Instrum. Meth. **A425** 357 (1999).

[7] F. James, 'Minuit Reference Manual, Version 94.1', CERN Program Library Long Writeup D506 (1994). See also the `root` documentation for the class `TMinuit` and `http:\\root.cern.ch`

[8] W. A. Rolke, A. M. López, J. Conrad, 'Limits and Confidence Intervals in the Presence of Nuisance Parameters', `arXiv:physics/0403059` (2004).

[9] T. Junk, 'Building a More General χ^2', CDF 7904 (2005).

[10] A. L. Read, J. Phys. G **28**, 2693 (2002).
T. Junk, Nucl. Instrum. Meth. A **434**, 435 (1999).

[11] R. Barlow, eConf **C030908**, WEMT002 (2003) [arXiv:physics/0401042].

[12] J. Heinrich, "Bayesian limit software: multi-channel with correlated backgrounds and efficiencies", CDF 7587 (2005).